B 9

```
NNN         NNN  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT     AAAAAAAAA        CCCCCCCCCCCC  PPPPPPPPPPPP
NNN         NNN  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT     AAAAAAAAA        CCCCCCCCCCCC  PPPPPPPPPPPP
NNN         NNN  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT     AAAAAAAAA        CCCCCCCCCCCC  PPPPPPPPPPPP
NNN         NNN  EEE                    TTT        AAA         AAA  CCC              PPP         PPP
NNN         NNN  EEE                    TTT        AAA         AAA  CCC              PPP         PPP
NNN         NNN  EEE                    TTT        AAA         AAA  CCC              PPP         PPP
NNNNNN      NNN  EEE                    TTT        AAA         AAA  CCC              PPP         PPP
NNNNNN      NNN  EEE                    TTT        AAA         AAA  CCC              PPP         PPP
NNNNNN      NNN  EEE                    TTT        AAA         AAA  CCC              PPP         PPP
NNN   NNN   NNN  EEEEEEEEEEEE           TTT        AAA         AAA  CCC              PPPPPPPPPPPP
NNN   NNN   NNN  EEEEEEEEEEEE           TTT        AAA         AAA  CCC              PPPPPPPPPPPP
NNN   NNN   NNN  EEEEEEEEEEEE           TTT        AAA         AAA  CCC              PPPPPPPPPPPP
NNN      NNNNNN  EEE                    TTT        AAAAAAAAAAAAAAA  CCC              PPP
NNN      NNNNNN  EEE                    TTT        AAAAAAAAAAAAAAA  CCC              PPP
NNN      NNNNNN  EEE                    TTT        AAAAAAAAAAAAAAA  CCC              PPP
NNN         NNN  EEE                    TTT        AAA         AAA  CCC              PPP
NNN         NNN  EEE                    TTT        AAA         AAA  CCC              PPP
NNN         NNN  EEE                    TTT        AAA         AAA  CCC              PPP
NNN         NNN  EEEEEEEEEEEEEEE        TTT        AAA         AAA     CCCCCCCCCCCC  PPP
NNN         NNN  EEEEEEEEEEEEEEE        TTT        AAA         AAA     CCCCCCCCCCCC  PPP
NNN         NNN  EEEEEEEEEEEEEEE        TTT        AAA         AAA     CCCCCCCCCCCC  PPP
```

```
**FILE**ID**NETEVTLOG                              M  8









NN      NN EEEEEEEEEE TTTTTTTTTT EEEEEEEEEE VV      VV TTTTTTTTTT LL           OOOOOO     GGGGGGGG
NN      NN EEEEEEEEEE TTTTTTTTTT EEEEEEEEEE VV      VV TTTTTTTTTT LL           OOOOOO     GGGGGGGG
NN      NN EE             TT     EE         VV      VV     TT     LL         OO      OO GG
NN      NN EE             TT     EE         VV      VV     TT     LL         OO      OO GG
NNNN    NN EE             TT     EE         VV      VV     TT     LL         OO      OO GG
NNNN    NN EE             TT     EE         VV      VV     TT     LL         OO      OO GG
NN  NN  NN EEEEEEEE       TT     EEEEEEEE   VV      VV     TT     LL         OO      OO GG
NN  NN  NN EEEEEEEE       TT     EEEEEEEE   VV      VV     TT     LL         OO      OO GG
NN    NNNN EE             TT     EE         VV      VV     TT     LL         OO      OO GG  GGGGGG
NN    NNNN EE             TT     EE         VV      VV     TT     LL         OO      OO GG  GGGGGG
NN      NN EE             TT     EE           VV  VV       TT     LL         OO      OO GG      GG    ....
NN      NN EE             TT     EE           VV  VV       TT     LL         OO      OO GG      GG    ....
NN      NN EEEEEEEEEE     TT     EEEEEEEEEE     VV         TT     LLLLLLLLLL   OOOOOO     GGGGGG      ....
NN      NN EEEEEEEEEE     TT     EEEEEEEEEE     VV         TT     LLLLLLLLLL   OOOOOO     GGGGGG      ....

LL           IIIIII     SSSSSSSS
LL           IIIIII     SSSSSSSS
LL             II     SS
LL             II     SS
LL             II     SS
LL             II     SS
LL             II       SSSSSS
LL             II       SSSSSS
LL             II             SS
LL             II             SS
LL             II             SS
LL             II             SS
LLLLLLLLLL   IIIIII   SSSSSSSS
LLLLLLLLLL   IIIIII   SSSSSSSS
```

```
0000         .TITLE  NETEVTLOG - Process Event logging needs
0000         .IDENT  'V04-000'
0000         .DEFAULT DISPLACEMENT,WORD
0000
0000  ;****************************************************************************
0000  ;*                                                                          *
0000  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000  ;*  ALL RIGHTS RESERVED.                                                    *
0000  ;*                                                                          *
0000  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000  ;*  TRANSFERRED.                                                            *
0000  ;*                                                                          *
0000  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000  ;*  CORPORATION.                                                            *
0000  ;*                                                                          *
0000  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000  ;*                                                                          *
0000  ;*                                                                          *
0000  ;****************************************************************************
0000  ;
0000  ;
0000  ; FACILITY:      NETWORK ACP
0000  ;
0000  ; ABSTRACT:
0000  ;
0000  ;       This module performs the bulk of processing required to
0000  ;       take care of network event logging needs.
0000  ;
0000  ; ENVIRONMENT:
0000  ;
0000  ;       MODE = KERNEL
0000  ;
0000  ; AUTHOR:        Scott G. Davis, CREATION DATE: 03-JUL-1980
0000  ;
0000  ; MODIFIED BY:
0000  ;
0000  ;       V015    RNG0015         Rod Gamache     18-Jun-1984
0000  ;               Log Data Base Re-used events.
0000  ;
0000  ;       V014    TMH0014         Tim Halvorsen   28-Apr-1983
0000  ;               Make LDO use "Adjacent node" rather than
0000  ;               "Expected node".
0000  ;
0000  ;       V013    RNG0013         Rod Gamache     21-Apr-1983
0000  ;               Save/Restore R1 in NET$DBC_EFI/ESI.
0000  ;
0000  ;       V012    TMH0012         Tim Halvorsen   07-Apr-1983
0000  ;               Allow caller to specify that no REASON parameter is to
0000  ;               be logged on TPL events.
0000  ;
```

```
0000     ;         V011     TMH0011         Tim Halvorsen   29-Mar-1983
0000     ;                  Add "aborted service request".
0000     ;
0000     ;         V010     TMH0010         Tim Halvorsen   22-Dec-1982
0000     ;                  Modify a number of events to log packet beginning
0000     ;                  (first 16 bytes) rather than packet header, which
0000     ;                  implies a Phase III route header.
0000     ;
0000     ;         V009     TMH0009         Tim Halvorsen   05-Nov-1982
0000     ;                  Add code to suppress the area number in node
0000     ;                  addresses, if area routing is being hidden.
0000     ;                  Fix area reachability chage so that it reports
0000     ;                  the source area, not the source node.
0000     ;
0000     ;         V008     TMH0008         Tim Halvorsen   16-Sep-1982
0000     ;                  Add support for automatic line counters.
0000     ;
0000     ;         V007     TMH0007         Tim Halvorsen   27-Jul-1982
0000     ;                  Add support to handle Phase IV events.
0000     ;                  Increase size of event buffer to allow for a large
0000     ;                  number of "node reachable" events at startup time.
0000     ;                  Rewrite READ_EVENTS so that it correctly transfers
0000     ;                  only as many WHOLE events as will fit into the caller's
0000     ;                  buffer, and so that it correctly shifts the remaining
0000     ;                  events in the event buffer correctly.  The previous
0000     ;                  code was returning partial event records to EVL, and
0000     ;                  causing the number of bytes "left" to be incorrectly
0000     ;                  computed to be a very small number, thus wasting most
0000     ;                  of the event buffer.
0000     ;
0000     ;         V006     TMH0006         Tim Halvorsen   30-Jun-1982
0000     ;                  Dynamically allocate event buffer, rather than having
0000     ;                  it statically defined in impure own storage.
0000     ;                  Remove all explicit addressing specifiers, and make
0000     ;                  the default addressing = word for the entire module.
0000     ;
0000     ;         V005     TMH0005         Tim Halvorsen   12-Apr-1982
0000     ;                  Get address of utility buffer from cell, rather than
0000     ;                  referencing a statically defined location.
0000     ;                  Fix STARTUP_EVL to queue a WQE to do the job, since
0000     ;                  STARTUP_OBJ calls CNF action routines, some of which
0000     ;                  allocate the CNF static temporary buffer.  Unfortunately,
0000     ;                  this is required because we may be logging counters while
0000     ;                  having the static temporary buffer allocated (CNT does this).
0000     ;                  Fix code to search database using FNDMIN operator to expect
0000     ;                  that the matched CNF will be returned in R10.
0000     ;                  Fix bug which prevented the node address from being shown
0000     ;                  in the event display if there is no node name currently
0000     ;                  associated with that address.
0000     ;                  Fix bug in reporting of "packet format error" event which
0000     ;                  showed garbage after "packet beginning" parameter.
0000     ;                  Add formatting for "local node state change",
0000     ;                  "locally initiated state change", and "remotely initiated
0000     ;                  state change" events.
0000     ;
0000     ;         V03-04   ADE0025         A.Eldridge      01-Feb-1981
0000     ;                  Remove parameter count in front of counter block.
```

```
0000   ;                Fix database determination while processing the counter timer.
0000   ;
0000   ;        V03-03  ADE0024         A.Eldridge      19-Jan-1981
0000   ;                Include the "packet beginning" and not the "packet header"
0000   ;                as part of the event data for circuit initialization failure
0000   ;                events.
0000   ;
0000   ;        V03-02  ADE0023         Al Eldridge     30-Nov-1981
0000   ;                Added zero counter event.
0000   ;
0000   ;        V03-01                  Al Eldridge     01-Nov-1981
0000   ;                Udgrade to V3.0.0 Network management.  The changes are
0000   ;                primarily related to the change to the new Circuit/Line
0000   ;                model of the datalink layer.
0000   ;
0000   ;        V022    ADE0022         Al Eldridge     05-Sep-1980
0000   ;                Further fixes to counter logging.
0000   ;
0000   ;        V021    TMH0021         Tim Halvorsen   04-Sep-1980
0000   ;                Pass null string as SYS$NET to EVL process.  Preserve all
0000   ;                registers in NET$DBC_EFI,ESI.  Remove temporary definition of
0000   ;                EVC$C_VMS_DBC (was decimal 2000, should be hex 2000) and use
0000   ;                $EVCDEF.
0000   ;
0000   ;        V020    ADE0020         Al Eldridge     20-Aug-1980
0000   ;                Log internally detected events. Log counters.
0000   ;
```

```
                    0000         .SBTTL  DECLARATIONS
                    0000   ;
                    0000   ; MACROS
                    0000   ;
                    0000           $ADJDEF
                    0000           $MSGDEF
                    0000           $NETSYMDEF
                    0000           $NETUPDDEF
                    0000           $NFBDEF
                    0000           $CNFDEF
                    0000           $CNRDEF
                    0000           $PRVDEF
                    0000           $RCBDEF
                    0000           $NMADEF
                    0000           $EVCDEF
                    0000           $RAWDEF
                    0000           $WQEDEF
                    0000
                    0000   ;
                    0000   ; EQUATED SYMBOLS:
                    0000   ;
                    0000
          000000C1  0000   NMA$C_PTY_CM1    = 193                     ; && until it gets added to $NMADEF
                    0000
                    0000
          0000001A  0000   EVL_OBJ          = 26                      ; Event logger object number
          00000005  0000   NET$C_EVTTHRESH = 5                        ; Event threshold
          02FAF080  0000   NET$C_EVTTIMER  = 10*1000*1000*5           ; Timer constant
          00001F40  0000   NET$C_EVTBUFLTH = 8000                     ; Length of event buffer
          00000020  0000   NET$C_LSTEVTLTH = 32                       ; Length of "lost event" event
          00001F00  0000   NET$C_AVLBUFLTH = NET$C_EVTBUFLTH -        ; Length for normal events
                    0000                   - <2*NET$C_LSTEVTLTH>
                    0000
                    0000   ;
                    0000   ; mailbox message mask definitions
                    0000   ;
                    0000
          00000001  0000   MBX$V_EVTAVL     = 1                       ; Mask bit for MSG$_EVTAVL
          00000002  0000   MBX$V_EVTRCVCHG = 2                        ; Mask bit for MSG$_EVTRCVCHG
          00000003  0000   MBX$V_EVTXMTCHG = 3                        ; Mask bit for MSG$_EVTXMTCHG
```

```
                    0000   ;
                    0000   ; OWN STORAGE:
                    0000   ;
            00000000               .PSECT  NET_IMPURE,WRT,NOEXE,LONG
                    0000
  00000000  0000   CNX_PLI_OLDTIM: .LONG   0                          ; Old CNF timer for PLI's
  00000000  0004   CNX_CRI_OLDTIM: .LONG   0                          ; Old CNF timer for CRI's
  00000000  0008   CNX_NDI_OLDTIM: .LONG   0                          ; Old CNF timer for NDI's
                    000C
       01'  000C   EVT_B_FLAGS:    .BYTE   EVT$M_EVTAVL               ; Allow immediate event message
                    000D
                    000D           $VIELD  EVT,0,-                    ; Define the flags
                    000D           <-
                    000D               <EVTAVL,1,M>,-      ; Flag implies MSG$_EVTAVL can be sent
                    000D               <LOSTEVENT,1,M>,-   ; Flag implies "lost event" event occurred
                    000D               <DBCEVENT,1,M>,-    ; Database change event logged
                    000D               <CST_PLI,1,M>,-     ; Line counter suppression timer ticking
                    000D               <CST_CRI,1,M>,-     ; Circuit counter suppression timer ticking
                    000D               <CST_NDI,1,M>,-     ; Node counter suppression timer ticking
                    000D           >
                    000D
  0000000F  000D   EVT_W_THRESH:   .BLKW   1                          ; No. of events available
      0000  000F   EVT_W_LOST:     .WORD   0                          ; # event bytes lost
      0000  0011   EVT_W_PEAK:     .WORD   0                          ; Peak value of EVT_W_LOST
  00000000  0013   BASE_TIME:      .LONG   0                          ; Base time for counter logging
                    0017
                    0017           .ALIGN  LONG
  00000000  0018   EVT_L_BUFFER:   .LONG   0                          ; Address of event buffer
  00000000  001C   EVT_L_BUFPTR:   .LONG   0                          ; Ptr to next buffer location
                    0020
                    0020   LOST_EVENT:                                ; Block to hold "lost event"
     001E'  0020                   .WORD   10$-LOST_EVENT             ; Length of event
  0000002A  0022                   .BLKQ   1                          ; For time-stamp
      0000  002A                   .WORD   EVC$C_NMA_LOS              ; Event code
        FF  002C                   .BYTE   -1                         ; No source for this event
  0000003E  002D                   .BLKB   17                         ; No event-ID
                    003E   10$:
                    003E
                    003E   DBC_EVENT:                                 ; Block containing "DBC event"
     001E'  003E                   .WORD   10$-DBC_EVENT              ; Length of event
  00000048  0040                   .BLKQ   1                          ; For time-stamp
      2000  0048                   .WORD   EVC$C_VMS_DBC              ; Event code
        FF  004A                   .BYTE   -1                         ; No source for this event
  0000005C  004B                   .BLKB   17                         ; No event-ID
                    005C   10$:
                    005C
                    005C   NET$AB_EVT_WQE::
  00000080  005C                   .BLKB   WQE$C_LENGTH               ; Common WQE for event reporting
                    0080
                    0080
            00000000               .PSECT  NET_PURE,LONG,NOWRT,NOEXE
                    0000
                    0000
  00000000  0000                   CNX$B_SPARE     =  0    ; Spare, reserved for future use
  00000001  0000                   CNX$B_TIM_SUP   =  1    ; RCB suppression timer bit i.d.
  00000002  0000                   CNX$W_ID_CTM    =  2    ; WQE timer REQIDT field and database i.d.
  00000004  0000                   CNX$L_COUNTER   =  4    ; CNF field i.d. of counter string
```

```
00000008  0000   241            CNX$L_DEL_TIME  =  8     ; CNF field i.d. of delta timer value
0000000C  0000   242            CNX$L_ABS_TIME  =  12    ; CNF field i.d. of absolute timer value
00000010  0000   243            CNX$L_OLD_TIME  =  16    ; Ptr to oldest CNF absolute due time value
00000014  0000   244            CNX$L_CNR_PTR   =  20    ; Ptr to CNR pointer
00000018  0000   245            CNX$C_LENGTH    =  24
          0000   246
          0000   247 CNX_PLI:                            ; PLI CNX
      00  0000   248            .BYTE   0                ; Spare
      03  0001   249            .BYTE   evt$v_cst_pli    ; Log datalink counter suppression timer id
    0001  0002   250            .WORD   evc$c_src_lin    ; WQE REQIDT value for datalinks
          0004   251            .CNFFLD pli,s,cnt        ; Datalink counter string field i.d.
          0008   252            .CNFFLD pli,l,lct        ; Datalink counter timer field i.d.
          000C   253            .CNFFLD pli,l,cta        ; Datelink absolute timer field i.d.
00000000' 0010   254            .LONG   cnx_pli_oldtim   ; Due time of oldest CNFs
00000000' 0014   255            .LONG   net$gl_cnr_pli   ; Address of CRI CNR pointer
          0018   256
          0018   257 CNX_CRI:                            ; CRI CNX
      00  0018   258            .BYTE   0                ; Spare
      04  0019   259            .BYTE   evt$v_cst_cri    ; Log datalink counter suppression timer id
    0003  001A   260            .WORD   evc$c_src_cir    ; WQE REQIDT value for datalinks
          001C   261            .CNFFLD cri,s,cnt        ; Datalink counter string field i.d.
          0020   262            .CNFFLD cri,l,lct        ; Datalink counter timer field i.d.
          0024   263            .CNFFLD cri,l,cta        ; Datelink absolute timer field i.d.
00000004' 0028   264            .LONG   cnx_cri_oldtim   ; Due time of oldest CNFs
00000000' 002C   265            .LONG   net$gl_cnr_cri   ; Address of CRI CNR pointer
          0030   266
          0030   267 CNX_NDI:                            ; NDI CNX
      00  0030   268            .BYTE   0                ; Spare
      05  0031   269            .BYTE   evt$v_cst_ndi    ; Log node counter suppression timer id
    0000  0032   270            .WORD   evc$c_src_nod    ; WQE REQIDT value for nodes
          0034   271            .CNFFLD ndi,s,cnt        ; Node counter string field i.d.
          0038   272            .CNFFLD ndi,l,cti        ; Node counter timer field i.d.
          003C   273            .CNFFLD ndi,l,cta        ; Node absolute timer field i.d.
00000008' 0040   274            .LONG   cnx_ndi_oldtim   ; Due time of oldest CNFs
00000000' 0044   275            .LONG   net$gl_cnr_ndi   ; Address of NDI CNR pointer
          0048   276
```

```
                    00000000  278           .PSECT  NET_CODE,NOWRT,LONG,EXE
                        0000  279
                        0000  280           .SBTTL  Event timer action routine
                        0000  281 ;+
                        0000  282 ; EVT_TIMER - This routine is called when the event timer threshold expires.
                        0000  283 ;
                        0000  284 ; FUNCTIONAL DESCRIPTION:
                        0000  285 ;
                        0000  286 ; Set the EVENT AVAILABLE flag (NET$V_EVTAVL)
                        0000  287 ;
                        0000  288 ;-
                        0000  289 EVT_TIMER:
              55    DD  0000  290           PUSHL   R5                      ; Save timer block address
              01    88  0002  291           BISB2   #EVT$M_EVTAVL,-         ; Set the flag
         000C'CF        0004  292                           EVT_B_FLAGS     ;
         000D'CF    B5  0007  293           TSTW    EVT_W_THRESH            ; Any events?
              03    13  000B  294           BEQL    10$                     ; If EQL no msgs, yet
            0378    30  000D  295           BSBW    SEND_EVT_MSG            ; Send MBX MSG
              50  8ED0  0010  296 10$:      POPL    R0                      ; Recover timer block
           FFEA'    30  0013  297           BSBW    WQE$DEALLOCATE          ; Deallocate it
              05        0016  298           RSB                             ; Done
```

```
                           0017        .SBTTL  Internal inbound raw event processing
                           0017 ;+
                           0017 ; NET$EVT_INTRAW - Process raw event detected internally
                           0017 ;
                           0017 ; FUNCTIONAL DESCRIPTION:
                           0017 ;
                           0017 ; A raw event is passed internally via a WQE.  It is formatted and put into
                           0017 ; the event buffer.
                           0017 ;
                           0017 ; INPUTS:        R11     CNR pointer as appropriate
                           0017 ;                R10     CNF pointer as appropriate
                           0017 ;                R9-R7   Scratch
                           0017 ;                R6      LPD pointer if datalink event
                           0017 ;                        XWB pointer if logical link event
                           0017 ;                        else srcatch
                           0017 ;                R5      WQE pointer if approriate
                           0017 ;
                           0017 ; OUTPUTS:       All registers are preserved
                           0017 ;
                           0017 ;-
                           0017 NET$EVT_INTRAW::                             ; Process internal raw event
           FFE6'   30      0017        BSBW    NET$GETUTLBUF            ; Get permission to use the utility
                           001A                                         ; buffer (co-routine call)
                           001A
       0FFF 8F     BB      001A        PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                           001E
53  02  0000'CF    C1      001E        ADDL3   NET$GL_UTLBUF,#2,R3      ; Setup output pointer
 83  00000000'GF   7D      0024        MOVQ    G^EXE$GQ_SYSTIME,(R3)+   ; Enter standard quadword time
     50   1C A5    3C      002B        MOVZWL  WQE$W_EVL_CODE(R5),R0    ; Get the raw event code
        83    50   B0      002F        MOVW    R0,(R3)+                 ; Enter the code
              17   10      0032        BSBB    50$                      ; Dispatch to complete building the
                           0034                                         ; event
           0F 50   E9      0034        BLBC    R0,40$                   ; If LBC then abort logging
      58  0000'CF  D0      0037        MOVL    NET$GL_UTLBUF,R8         ; Get original output pointer
      57  53   58  C3      003C        SUBL3   R8,R3,R7                 ; Calculate the data length
          68   57  B0      0040        MOVW    R7,(R8)                  ; Store as the length field
            0290   30      0043        BSBW    INTERNAL_EVENT           ; Stuff it into the event buffer
                           0046
       0FFF 8F     BA      0046 40$:   POPR    #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                   05      004A        RSB
                           004B
                           004B 50$:   ;
                           004B        ;  Dispatch to finish building the event.  The $DISPATCH skip chain is
                           004B        ;  used instead of one large $DISPATCH since the index codes are closely
                           004B        ;  packed within a class but widely separated from one class to another.
                           004B        ;
                           004B        $DISPATCH  R0,-
                           004B        <-
                           004B            <EVC$C_TPL_APL,  NON_PKT>,  -; Aged packet loss
                           004B            <EVC$C_TPL_UPL,  CIR_PKT>,  -; Unreachable packet loss
                           004B            <EVC$C_TPL_RPL,  CIR_PKT>,  -; Out-of-range packet loss
                           004B            <EVC$C_TPL_OPL,  CIR_PKT>,  -; Oversized packet loss
                           004B            <EVC$C_TPL_PFM,  CIR_BEG>,  -; Packet format error
                           004B            <EVC$C_TPL_PRU,  PRU>       -; Partial routing update loss
                           004B            <EVC$C_TPL_VFR,  VFR>,      -; Verification reject
                           004B            <EVC$C_TPL_LDO,  LDO>,      -; Circuit down, operator fault
                           004B            <EVC$C_TPL_LDS,  LDS>,      -; Circuit down, software fault
```

```
                    004B                <EVC$C_TPL_LDF,  LDF>,        -; Circuit down, Circuit fault
                    004B                <EVC$C_TPL_LUP,  CIR_ADJ>,    -; Circuit up
                    004B                <EVC$C_TPL_IOF,  IOF>,        -; Init failed, operator fault
                    004B                <EVC$C_TPL_ISF,  ISF>,        -; Init failed, software fault
                    004B                <EVC$C_TPL_ILF,  ILF>,        -; Init failed, Circuit fault
                    004B                <EVC$C_TPL_RCH,  RCH>,        -; Node reachability change
                    004B                <EVC$C_TPL_AUP,  CIR_ADJ>,    -; Adjacency up
                    004B                <EVC$C_TPL_ARJ,  CIR_ADJ>,    -; Adjacency rejected
                    004B                <EVC$C_TPL_ACH,  ACH>         -; Area reachibility change
                    004B            >
                    0079
                    0079            $DISPATCH  R0,-
                    0079            <-
                    0079                <EVC$C_NMA_CTR,  COUNTER>,    -; Automatic counter timer
                    0079                <EVC$C_NMA_ZER,  COUNTER>,    -; NCP ZERO counters command
                    0079                <EVC$C_NMA_ABS,  ABS>,        -; Aborted service request
                    0079            >
                    0083
                    0083            $DISPATCH  R0,-
                    0083            <-
                    0083                <EVC$C_NSL_DBR,  COUNTER>,    -; Data base re-used event
                    0083            >
                    008B
                    008B            $DISPATCH  R0,-
                    008B            <-
                    008B                <EVC$C_SCL_LNS,  LNS>,        -; Local node state change
                    008B            >
                    0093
                    0093            $DISPATCH  R0,-
                    0093            <-
                    0093                <EVC$C_DLL_LSC,  LSC>,        -; Locally initated state change
                    0093                <EVC$C_DLL_RSC,  RSC>,        -; Remotely initiated state change
                    0093            >
          50   D4   009D            CLRL     R0                        ; Event unknown
               05   009F            RSB                                ; Done
                    00A0
                    00A0  COUNTER:
   50  1E A5   9A   00A0            MOVZBL   WQE$B_EVL_DT1(R5),R0      ; Get counter database i.d.
                    00A4            $DISPATCH  R0,-                    ; Dispatch on database type
                    00A4            <-
                    00A4                <EVC$C_SRC_NOD,  NOD_COU>,-   ; Log and clear node counters
                    00A4                <EVC$C_SRC_CIR,  CIR_COU>,-   ; Log and clear circuit counters
                    00A4                <EVC$C_SRC_LIN,  LIN_COU>,-   ; Log and clear line counters
                    00A4            >
          50   D4   00B0            CLRL     R0                        ; Database unknown
               05   00B2            RSB                                ; Done
                    00B3
                    00B3  NOD_COU:                                     ; Node counters
        011B   30   00B3            BSBW     ENTER_SRCNOD              ; Enter source node i.d.
          08   11   00B6            BRB      COU                       ; Enter the counters
                    00B8  CIR_COU:                                     ; Circuit counters
        013A   30   00B8            BSBW     ENTER_SRCCIR              ; Enter Circuit i.d.
          03   11   00BB            BRB      COU                       ; Enter the counters
                    00BD  LIN_COU:                                     ; Line counters
        0154   30   00BD            BSBW     ENTER_SRCLIN              ; Enter Line ID
                    00C0
                    00C0  COU:                                         ; Log and clear the counters
```

```
                 55   DD  00C0              PUSHL   R5                          ; Save reg
         50   1F A5   9A  00C2              MOVZBL  WQE$B_EVL_DT2(R5),R0        ; Get length of counter block
   63    18 B5    50  28  00C6              MOVC3   R0,-                        ;
                          00CB                      @WQE$L_EVL_PKT(R5),(R3)     ; Move the counter block
         50      01   D0  00CB              MOVL    #1,R0                       ; Indicate success
                 55 8ED0  00CE              POPL    R5                          ; Recover WQE pointer
                      05  00D1              RSB
                          00D2
                          00D2      NON_PKT:                                    ; No source, packet header
               00DE   30  00D2              BSBW    ENTER_NO_SRC                ; Enter null source field
               0148   31  00D5              BRW     ENTER_PKTHDR                ; Enter the packet header
                          00D8
                          00D8      CIR_PKT:                                    ; Circuit source, adj, packet header
               011A   30  00D8              BSBW    ENTER_SRCCIR                ; Enter the source Circuit
            83   08   B0  00DB              MOVW    #EVC$C_TPL_PADJ,(R3)+       ; Identify next field
               0181   30  00DE              BSBW    PNA_NODE                    ; Enter partner node id
               013C   31  00E1              BRW     ENTER_PKTHDR                ; Enter the packet header
                          00E4
                          00E4      CIR_BEG:                                    ; Circuit source, adj, packet begining
               010E   30  00E4              BSBW    ENTER_SRCCIR                ; Enter the source Circuit
            83   08   B0  00E7              MOVW    #EVC$C_TPL_PADJ,(R3)+       ; Identify next field
               0175   30  00EA              BSBW    PNA_NODE                    ; Enter partner node id
               0159   31  00ED              BRW     ENTER_PPKB                  ; Enter packet begining
                          00F0
                          00F0      PRU:                                        ; Partial routing update loss
               0102   30  00F0              BSBW    ENTER_SRCCIR                ; Enter source Circuit
               0153   30  00F3              BSBW    ENTER_PPKB                  ; Enter the packet header
            83   02   B0  00F6              MOVW    #EVC$C_TPL_PHIA,(R3)+       ; Identify next field
            83   02   90  00F9              MOVB    #NMA$C_PTY_DU2,(R3)+        ; Identify field format
                          00FC
                          00FC              ASSUME  WQE$B_EVL_DT2-WQE$B_EVL_DT1 EQ 1
                          00FC
         83   1E A5   B0  00FC              MOVW    WQE$B_EVL_DT1(R5),(R3)+     ; Enter partner's highest
                          0100                                                  ; reachable node address
            83   08   B0  0100              MOVW    #EVC$C_TPL_PADJ,(R3)+       ; Identify adjacent node
               015C   30  0103              BSBW    PNA_NODE                    ; Enter partner node id
            50   01   90  0106              MOVB    #1,R0                       ; Success
                      05  0109              RSB
                          010A      VFR:                                        ; Verification reject
               00E8   30  010A              BSBW    ENTER_SRCCIR                ; Enter the source Circuit
            83   03   B0  010D              MOVW    #EVC$C_TPL_PNOD,(R3)+       ; Identify next field
               014F   31  0110              BRW     PNA_NODE                    ; Enter partner node id
                          0113
                          0113      IOF:                                        ; Init failure, operator fault
                 23   10  0113              BSBB    ISF                         ; Same as ISF, except add:
            83   06   B0  0115              MOVW    #EVC$C_TPL_PVRS,(R3)+       ; Identify next field (version)
         83   C3 8F   90  0118              MOVB    #NMA$C_PTY_CM3,(R3)+        ; Enter format type
            83   01   90  011C              MOVB    #NMA$C_PTY_DU1,(R3)+        ; Enter format type
      83   0000'CF    90  011F              MOVB    NET$GL_INITVER,(R3)+        ; Enter version number
            83   01   90  0124              MOVB    #NMA$C_PTY_DU1,(R3)+        ; Enter format type
      83   0001'CF    90  0127              MOVB    NET$GL_INITVER+1,(R3)+      ; Enter ECO number
            83   01   90  012C              MOVB    #NMA$C_PTY_DU1,(R3)+        ; Enter format type
      83   0002'CF    90  012F              MOVB    NET$GL_INITVER+2,(R3)+      ; Enter user ECO number
            50   01   90  0134              MOVB    #1,R0                       ; Success
                      05  0137              RSB
                          0138      ISF:                                        ; Init failure, software fault
                 17   10  0138              BSBB    CIR_REASON                  ; Enter circuit id, reason
```

```
          010C   31  013A             BRW     ENTER_PPKB                ; Enter packet header
                     013D
                     013D  LDO:
                     013D  LDS:                                         ; Adjacency forced down by software
            12   10  013D             BSBB    CIR_REASON                ; Enter common header
       83   08   B0  013F             MOVW    #EVC$C_TPL_PADJ,(R3)+     ; Identify next field
          011D   30  0142             BSBW    PNA_NODE                  ; Enter partner node id
          0101   31  0145             BRW     ENTER_PPKB                ; Enter packet header
                     0148
                     0148  CIR_ADJ:                                     ; Enter circuit id, adjacent node
          00AA   30  0148             BSBW    ENTER_SRCCIR              ; Enter source Circuit id
       83   08   B0  014B             MOVW    #EVC$C_TPL_PADJ,(R3)+     ; Identify adjacent node
          0111   31  014E             BRW     PNA_NODE                  ; Enter partner node id
                     0151
                     0151  ILF:                                         ; Init failure, circuit fault
                     0151  LDF:                                         ; Circuit failure, Circuit fault
                     0151  CIR_REASON:                                  ; Enter circuit id, reason code
          00A1   30  0151             BSBW    ENTER_SRCCIR              ; Enter source Circuit id
         1E A5   95  0154             TSTB    WQE$B_EVL_DT1(R5)         ; Any reason specified?
            05   19  0157             BLSS    90$                       ; Exit if not
       83   05   B0  0159             MOVW    #EVC$C_TPL_PRSN,(R3)+     ; Identify next field
            12   11  015C             BRB     CD1                       ; Enter field's value
       50   01   90  015E  90$:       MOVB    #1,R0                     ; Signal success
                 05  0161             RSB
                     0162
                     0162  RCH:                                         ; Node reachability change
          006C   30  0162             BSBW    ENTER_SRCNOD              ; Enter the source node
       83   07   B0  0165             MOVW    #EVC$C_TPL_PSTS,(R3)+     ; Identify next field
            06   11  0168             BRB     CD1
                     016A
                     016A  ACH:                                         ; Area reachability change
          004E   30  016A             BSBW    ENTER_SRCAREA             ; Enter the source area
       83   07   B0  016D             MOVW    #EVC$C_TPL_PSTS,(R3)+     ; Identify next field
                     0170
    83   81 8F   90  0170  CD1:       MOVB    #NMA$C_PTY_CD1,(R3)+      ; Enter field format type
    83   1E A5   90  0174             MOVB    WQE$B_EVL_DT1(R5),(R3)+   ; Enter qualifying data byte
       50   01   90  0178             MOVB    #1,R0                     ; Signal success
                 05  017B             RSB
                     017C
          0034   30  017C  LNS:       BSBW    ENTER_NO_SRC              ; Enter no source ID
       83   00   B0  017F             MOVW    #EVC$C_SCL_PRSN,(R3)+     ; Enter "reason" parameter type
    83   81 8F   90  0182             MOVB    #NMA$C_PTY_CD1,(R3)+      ; Enter field format type
    83   18 A5   90  0186             MOVB    WQE$L_EVL_PKT(R5),(R3)+   ; Enter reason code
       83   01   B0  018A             MOVW    #EVC$C_SCL_POLD,(R3)+     ; Enter "old state" parameter type
            E1   10  018D             BSBB    CD1                       ; Enter coded byte from DT1
       83   02   B0  018F             MOVW    #EVC$C_SCL_PNEW,(R3)+     ; Enter "new state" parameter type
    83   81 8F   90  0192  CD1_2:     MOVB    #NMA$C_PTY_CD1,(R3)+      ; Enter field format type
    83   1F A5   90  0196             MOVB    WQE$B_EVL_DT2(R5),(R3)+   ; Enter qualifying data byte
       50   01   90  019A             MOVB    #1,R0                     ; Signal success
                 05  019D             RSB
                     019E
                     019E  LSC:
          0054   30  019E  RSC:       BSBW    ENTER_SRCCIR              ; Enter source circuit
       83   00   B0  01A1             MOVW    #EVC$C_DLL_POLD,(R3)+     ; Enter "old state" parameter type
            CA   10  01A4             BSBB    CD1                       ; Enter coded byte from DT1
       83   01   B0  01A6             MOVW    #EVC$C_DLL_PNEW,(R3)+     ; Enter "new state" parameter type
            E7   11  01A9             BRB     CD1_2                     ; Enter coded byte from DT2; and exit
```

```
                       01AB   528
                       01AB   529 ABS:                                      ; "Aborted service request"
                       01AB   530                                           ; Enter circuit id, reason code
          0047   30    01AB   531          BSBW    ENTER_SRCCIR             ; Enter source Circuit id
       83   03   B0    01AE   532          MOVW    #EVC$C_NMA_PRSN,(R3)+    ; Identify next field
            BD   11    01B1   533          BRB     CD1                      ; Enter field's value
```

```
                                 01B3   535 ENTER_NO_SRC:                                   ; Enter null source field
              83   FF 8F   90    01B3   536           MOVB    #EVC$C_SRC_NON,(R3)+          ; No source
                      57   D4    01B7   537           CLRL    R7                            ; Init count field
                      47   11    01B9   538           BRB     ENT_17                        ; Zero the source field
                                 01BB   539
                                 01BB   540 ENTER_SRCAREA:                                  ; Enter source area
              83      05   90    01BB   541           MOVB    #EVC$C_SRC_ARE,(R3)+          ; Enter source type
              83   12 A5   90    01BE   542           MOVB    WQE$W_REQIDT(R5),(R3)+        ; Store the area number
                      55   DD    01C2   543           PUSHL   R5                            ; Save registers
 63   10   00   6E    00   2C    01C4   544           MOVC5   #0,(SP),#0,#16,(R3)           ; Zero rest of 17 byte fixed field
                      55 8ED0    01CA   545           POPL    R5                            ; Restore registers
              50      01   D0    01CD   546           MOVL    #1,R0                         ; Success
                      05         01D0   547           RSB
                                 01D1   548
                                 01D1   549 ENTER_SRCNOD:                                   ; Enter source node
              83      00   90    01D1   550           MOVB    #EVC$C_SRC_NOD,(R3)+          ; Enter source type
         51        12 A5   3C    01D4   551           MOVZWL  WQE$W_REQIDT(R5),R1           ; Get the node address
                      09   12    01D8   552           BNEQ    10$                           ; Branch if not local node
         50      0000'CF   D0    01DA   553           MOVL    NET$GL_PTR_VCB,R0             ; Get the RCB address
         51        0E A0   3C    01DF   554           MOVZWL  RCB$W_ADDR(R0),R1             ; Enter the local node address
                   FE1A'   30    01E3   555 10$:      BSBW    SUPPRESS_AREA                 ; Suppress area, if necessary
              83      51   B0    01E6   556           MOVW    R1,(R3)+                      ; Enter the node address
                                 01E9   557           $CNFFLD ndi,s,nna,R9                  ; Identify the node name field
                      0D   10    01F0   558           BSBB    ENT_SRC                       ; Enter padded node name
                      73   B5    01F2   559           TSTW    -(R3)                         ; Backup two bytes to account for
                      05         01F4   560           RSB                                   ; node address at begining in order
                                 01F5   561                                                 ; to keep a total of 17 bytes
                                 01F5   562
                                 01F5   563 ENTER_SRCCIR:                                   ; Enter source Circuit id
              83      03   90    01F5   564           MOVB    #EVC$C_SRC_CIR,(R3)+          ; Enter source type
                                 01F8   565           $CNFFLD cri,s,nam,R9                  ; Get the Circuit name field i.d.
                   FDFE'   30    01FF   566 ENT_SRC:BSBW      CNF$GET_FIELD                 ; Get the source i.d. name
                      55   DD    0202   567 ENT_17:  PUSHL    R5                            ; Save critical reg
              83      57   90    0204   568           MOVB    R7,(R3)+                      ; Enter length of name
 63   10   00   68    57   2C    0207   569           MOVC5   R7,(R8),#0,#16,(R3)           ; Enter the name
                      55 8ED0    020D   570           POPL    R5                            ; Restore reg
              50      01   90    0210   571           MOVB    #1,R0                         ; Success
                      05         0213   572           RSB
                                 0214   573
                                 0214   574 ENTER_SRCLIN:                                   ; Enter source Line id
              83      01   90    0214   575           MOVB    #EVC$C_SRC_LIN,(R3)+          ; Enter source type
                                 0217   576           $CNFFLD pli,s,nam,R9                  ; Get the Line name field i.d.
                      DF   11    021E   577           BRB     ENT_SRC                       ; Store the parameter value
                                 0220   578
                                 0220   579 ENTER_PKTHDR:
         50        18 A5   D0    0220   580           MOVL    WQE$L_EVL_PKT(R5),R0          ; Get msg pointer
                      1F   13    0224   581           BEQL    90$                           ; Skip if none
              83      00   B0    0226   582           MOVW    #EVC$C_TPL_PPKH,(R3)+         ; Enter field i.d.
         83        C4 8F   90    0229   583           MOVB    #NMA$C_PTY_CM4,(R3)+          ; Format type for mulitple field
              83      21   90    022D   584           MOVB    #NMA$C_PTY_H1,(R3)+           ; Format type for message flags
              83      80   90    0230   585           MOVB    (R0)+,(R3)+                   ; Enter message flags
              83      02   90    0233   586           MOVB    #NMA$C_PTY_DU2,(R3)+          ; Format type for dst node
              83      80   B0    0236   587           MOVW    (R0)+,(R3)+                   ; Enter dst node address
              83      02   90    0239   588           MOVB    #NMA$C_PTY_DU2,(R3)+          ; Format type for src node
              83      80   B0    023C   589           MOVW    (R0)+,(R3)+                   ; Enter src node address
              83      21   90    023F   590           MOVB    #NMA$C_PTY_H1,(R3)+           ; Format type for visits field
              83      80   90    0242   591           MOVB    (R0)+,(R3)+                   ; Enter visits field
```

```
                                              B 10
NETEVTLOG                   - Process Event logging needs        16-SEP-1984 01:25:34  VAX/VMS Macro V04-00           Page  14
V04-000                     Internal inbound raw event processing  5-SEP-1984 02:20:54  [NETACP.SRC]NETEVTLOG.MAR;1          (6)

         50    01   90  0245   90$:    MOVB    #1,R0                       ; Success
                    05  0248           RSB
                        0249
                        0249   ENTER_PPKB:                                 ; Enter packet begining
         50  18 A5  D0  0249           MOVL    WQE$L_EVL_PKT(R5),R0        ; Get packet header pointer
               0F   13  024D           BEQL    90$                         ; Skip if none
         83    01   B0  024F           MOVW    #EVC$C_TPL_PPKB,(R3)+       ; Identify next field
         83    20   90  0252           MOVB    #NMA$C_PTY_HI,(R3)+         ; Enter format type
         83    10   90  0255           MOVB    #16,(R3)+                   ; Number of bytes to be entered
         83    80   7D  0258           MOVQ    (R0)+,(R3)+                 ; Enter first 8 bytes
         83    80   7D  025B           MOVQ    (R0)+,(R3)+                 ; Enter final 8 bytes
         50    01   90  025E   90$:    MOVB    #1,R0                       ; Success
                    05  0261           RSB
                        0262
                        0262   PNA_NODE:
         58  20 A5  3C  0262           MOVZWL  WQE$W_ADJ_INX(R5),R8        ; Get ADJ index
             FD97'  30  0266           BSBW    NET$FIND_ADJ                ; Find the associated ADJ
            35 50   E9  0269           BLBC    R0,50$                      ; If LBC then none found
         51  04 A7  3C  026C           MOVZWL  ADJ$W_PNA(R7),R1            ; Get the node address
               2F   13  0270           BEQL    50$                         ; If zero, then skip it
             FD8B'  30  0272           BSBW    SUPPRESS_AREA               ; Suppress area, if necessary
               2E   10  0275           BSBB    GET_NDI                     ; Find the NDI block
               57   95  0277           TSTB    R7                          ; Is there a node name ?
               0B   12  0279           BNEQ    5$                          ; If NEQ, then found
         83  C1 8F  90  027B           MOVB    #NMA$C_PTY_CM1,(R3)+        ; Enter only 1 field
         83    02   90  027F           MOVB    #NMA$C_PTY_DU2,(R3)+        ; Enter the address format type
         83    51   B0  0282           MOVW    R1,(R3)+                    ; Enter the address
                    05  0285           RSB                                 ; and skip the node name
         83  C2 8F  90  0286   5$:     MOVB    #NMA$C_PTY_CM2,(R3)+        ; Enter the complex format type
         83    02   90  028A           MOVB    #NMA$C_PTY_DU2,(R3)+        ; Enter the address format type
         83    51   B0  028D           MOVW    R1,(R3)+                    ; Enter the address
         83  40 8F  90  0290           MOVB    #NMA$C_PTY_AI,(R3)+         ; Enter the node name format type
         83    57   90  0294           MOVB    R7,(R3)+                    ; Enter the count field
         83    88   90  0297   10$:    MOVB    (R8)+,(R3)+                 ; Enter the text field
            FA 57   F5  029A           SOBGTR  R7,10$                      ;
         50    01   D0  029D           MOVL    #1,R0                       ; Indicate success
                    05  02A0           RSB
         53    02   C2  02A1   50$:    SUBL    #2,R3                       ; Remove parameter code
                    05  02A4           RSB
                        02A5
                        02A5   GET_NDI:
       0C02 8F      BB  02A5           PUSHR   #^M<R1,R10,R11>             ; Save regs
         58    51   D0  02A9           MOVL    R1,R8                       ; Copy node address
 5B    0000'CF      D0  02AC           MOVL    NET$GL_CNR_NDI,R11          ; Get NDI CNR
             FD4C'  30  02B1           BSBW    NET$NDI_BY_ADD              ; Find the NDI by address in R8
               57   7C  02B4           CLRQ    R7                          ; Nullify R7,R8
            0B 50   E9  02B6           BLBC    R0,10$                      ; No NDI CNF if LBC
                        02B9           $GETFLD ndi,s,nna                   ; Get the node name -- returns
                        02C4                                               ; R7,R8 = 0 if LBC in R0
       0C02 8F      BA  02C4   10$:    POPR    #^M<R1,R10,R11>             ; Restore regs
         50    01   D0  02C8           MOVL    #1,R0                       ; Report success (null node name is
                    05  02CB           RSB                                 ; okay)
```

```
                              02CC              .SBTTL  Inbound raw event processing
                              02CC      ;+
                              02CC      ; NET$LOG_EVENT - Put a raw event into the event buffer
                              02CC      ;
                              02CC      ; FUNCTIONAL DESCRIPTION:
                              02CC      ;
                              02CC      ; A raw event is passed to NETACP.  If a "lost event" event is already in
                              02CC      ; the raw event buffer, then the operation is ignored.  If there is no more
                              02CC      ; room for events, the "lost event" event is placed in the buffer and the
                              02CC      ; flag is set to so indicate.  If an event is placed in the buffer, and the
                              02CC      ; EVTAVL flag is set, then a mailbox message (MSG$_EVTAVL) is broadcast.
                              02CC      ; Events put into the buffer are time-stamped.
                              02CC      ;
                              02CC      ; INPUTS:          NET$GL_SIZ_P2 - size of input event
                              02CC      ;                  NET$GL_PTR_P2 - address of input event
                              02CC      ;
                              02CC      ; OUTPUTS:         MBX message may be broadcast (MSG$_EVTAVL)
                              02CC      ;                  R0 - Status
                              02CC      ;
                              02CC      ;-
                              02CC              .ENABL  LSB
                              02CC
                              02CC      NET$LOG_EVENT::                                 ; Entry point
           57    0000'CF  D0  02CC              MOVL    NET$GL_SIZ_P2,R7        ; Get no. of bytes in event
           58    0000'CF  D0  02D1              MOVL    NET$GL_PTR_P2,R8        ; Get address of event data
                              02D6
                              02D6      INTERNAL_EVENT:                         ; Local entry point
              68     57   B1  02D6              CMPW    R7,(R8)                 ; Counts must match
                     06   13  02D9              BEQL    5$                      ; If EQL OK
           50       00'   D0  02DB              MOVL    S^#SS$_BADPARAM,R0      ; Set error code
                   00C9   31  02DE              BRW     200$                    ; Take common exit
                              02E1              ;
                              02E1              ;   Ignore event if EFI database is empty (no events get transmitted)
                              02E1              ;
      2000 8F     0A A8   B1  02E1      5$:     CMPW    RAW$W_EVTCODE(R8),#EVC$C_VMS_DBC ; EFI database change
                     0A   13  02E7              BEQL    10$                     ; If so, buffer regardless of EFI list
           50    0000'CF  D0  02E9              MOVL    NET$GL_CNR_EFI,R0       ; Get address of EFI listhead
              60     50   D1  02EE              CMPL    R0,(R0)                 ; Is list empty?
                     3C   13  02F1              BEQL    14$                     ; If so, exit ignoring the event
                              02F3              ;
                              02F3              ;   If this is the first event to be buffered, then allocate an
                              02F3              ;   buffer to stored the event records until EVL picks them up.
                              02F3              ;
                0018'CF   D5  02F3      10$:    TSTL    EVT_L_BUFFER            ; Buffer allocated yet?
                     1A   12  02F7              BNEQ    11$                     ; Branch if so
     51    00001F4C 8F    D0  02F9              MOVL    #12+NET$C_EVTBUFLTH,R1  ; Set size of buffer needed
                  FCFD'   30  0300              BSBW    NET$ALLOCATE            ; Allocate the buffer
                  0D 50   E9  0303              BLBC    R0,11$                  ; If error, skip event reporting
    0018'CF       0C A2   9E  0306              MOVAB   12(R2),EVT_L_BUFFER     ; Store buffer pointer
  001C'CF       0018'CF   D0  030C              MOVL    EVT_L_BUFFER,EVT_L_BUFPTR ; Point to first available position
                              0313              ;
                              0313              ;   If "lost event" already reported, allow 1 data base change event
                              0313              ;   to get thru
                              0313              ;
                     01   E1  0313      11$:    BBC     #EVT$V_LOSTEVENT,-      ; If BC then try to buffer event
        20 000C'CF            0315                      EVT_B_FLAGS,20$
        000F'CF      57   A0  0319              ADDW    R7,EVT_W_LOST           ; Keep total of events lost
```

```
                0A A8    B1  031E   702          CMPW    RAW$W_EVTCODE(R8),-       ; No space - see if database change
              2000 8F        0321   703                  #EVC$C_VMS_DBC            ;
                   06    12  0324   704          BNEQ    12$                       ; If NEQ no - ignore event
                   02    E3  0326   705          BBCS    #EVT$V_DBCEVENT,-         ; If BC, database change not yet logged
         06 000C'CF          0328   706                  EVT_B_FLAGS,15$
                 007C    30  032C   707 12$:     BSBW    STARTUP_EVL               ; Start EVL process (if possible) in
                             032F   708                                            ; case it died and left our buffer full
                 0075    31  032F   709 14$:     BRW     100$                      ; Nothing to do
         58    003E'CF   9E  0332   710 15$:     MOVAB   DBC_EVENT,R8              ; Put in "DBC event" event
                   24    11  0337   711          BRB     25$                       ; Log the database change
                             0339   712 ;
                             0339   713 ;    If only room for one more event in buffer, insert "lost event"
                             0339   714 ;
  50   001C'CF   0018'CF     C3  0339   715 20$:     SUBL3   EVT_L_BUFFER,EVT_L_BUFPTR,R0    ; Compute # bytes in use
   50   00001F00 8F   50     C3  0341   716          SUBL3   R0,#NET$C_AVLBUFLTH,R0  ; Compute # bytes left
                   50    57  B1  0349   717          CMPW    R7,R0                     ; Enough space for this event?
                   12    1B  034C   718          BLEQU   30$                       ; If LEQU yes
           000F'CF   57  A0  034E   719          ADDW    R7,EVT_W_LOST             ; Keep total of events lost
                   02    88  0353   720          BISB2   #EVT$M_LOSTEVENT,-        ; Show that an event has been lost
              000C'CF        0355   721                  EVT_B_FLAGS               ;
         58    0020'CF   9E  0358   722          MOVAB   LOST_EVENT,R8             ; Put in "lost event" event
              57   68    3C  035D   723 25$:     MOVZWL  (R8),R7                   ; Get the length of the event
                             0360   724 ;
                             0360   725 ;    Insert event into buffer
                             0360   726 ;
           00000000'GF   7D  0360   727 30$:     MOVQ    G^EXE$GQ_SYSTIME,-        ; Time-stamp the event
                02 A8        0366   728                  RAW$T_SYSTIM(R8)
     001C'DF   68   57   28  0368   729          MOVC3   R7,(R8),@EVT_L_BUFPTR     ; Move event into the buffer
          001C'CF   53   D0  036E   730          MOVL    R3,EVT_L_BUFPTR           ; Update the pointer
              000D'CF    B6  0373   731          INCW    EVT_W_THRESH              ; Another event in buffer
                             0377   732 ;
                             0377   733 ;    If the event threshold has been reached, broadcast "events available" me
                             0377   734 ;
                   05    B1  0377   735          CMPW    #NET$C_EVTTHRESH,-        ; Has the threshold been reached?
              000D'CF        0379   736                  EVT_W_THRESH              ;
                   05    1E  037C   737          BGEQU   90$                       ; If GEQU no
                   01    88  037E   738          BISB2   #EVT$M_EVTAVL,-           ; Set the flag
              000C'CF        0380   739                  EVT_B_FLAGS               ;
                             0383   740
                             0383   741          ASSUME  EVT$V_EVTAVL EQ 0
                             0383   742
         1F 000C'CF      E9  0383   743 90$:     BLBC    EVT_B_FLAGS,100$          ; If LBC can't send mbx msg yet
                             0388   744
                             0388   745 ;
                             0388   746 ; It's OK to inform the world that the event buffer should be read
                             0388   747 ;
                             0388   748 SEND_EVT_MSG:
                             0388   749 ;
                             0388   750 ;    Startup EVL process if not already running
                             0388   751 ;
                   21    10  0388   752          BSBB    STARTUP_EVL               ; Startup EVL process if needed
                             038A   753 ;
                             038A   754 ;    Reset the threshold timer
                             038A   755 ;
                   51    D4  038A   756          CLRL    R1                        ; Set up REQIDT for canceling timer
          52    FC70 CF  9E  038C   757          MOVAB   EVT_TIMER,R2              ; Get action routine address for timer
  53   00000000 02FAF080 8F  7D  0391   758          MOVQ    #NET$C_EVTTIMER,R3        ; Let this much time elapse
```

```
                FC61'  30  039C   759           BSBW    WQE$RESET_TIM           ; Cancel previous timer, set new one
                           039F   760           ;
                           039F   761           ;   Now send the mailbox message
                           039F   762           ;
          53     02    D0  039F   763           MOVL    #<1@MBX$V_EVTAVL>,R3    ; Set mask
          52     3E    3C  03A2   764           MOVZWL  #MSG$_EVTAVL,R2         ; Set mbx msg code
                 43    10  03A5   765           BSBB    BROADCAST               ; Broadcast the message
          50     00'   3C  03A7   766 100$:     MOVZWL  S^#SS$_NORMAL,R0        ; Indicate success
                       05  03AA   767 200$:     RSB
                           03AB   768
                           03AB   769           .DSABL  LSB
```

```
                        03AB          .SBTTL  STARTUP_EVL - Start EVL process
                        03AB    ;+
                        03AB    ; STARTUP_EVL - Start EVL process
                        03AB    ;
                        03AB    ; Start EVL process (if possible).  This is done by queueing a WQE
                        03AB    ; to do the job, since STARTUP_OBJ calls CNF action routines, some
                        03AB    ; of which allocate the CNF static temporary buffer.  Unfortunately,
                        03AB    ; this is required because we may be logging counters while having
                        03AB    ; the static temporary buffer allocated (specifically, CNT does this).
                        03AB    ;
                        03AB    ; Inputs:
                        03AB    ;
                        03AB    ;       None
                        03AB    ;
                        03AB    ; Outputs:
                        03AB    ;
                        03AB    ;       None
                        03AB    ;
                        03AB    ;       R0 destroyed.
                        03AB    ;-
                        03AB
                        03AB    STARTUP_EVL:
          FC52'   30    03AB            BSBW    WQE$FORK                        ; Fork to work queue level
             52   7C    03AE            CLRQ    R2                              ; Pass nothing as SYS$NET to EVL
             54   7C    03B0            CLRQ    R4                              ; Use default process name
       58    1A   9A    03B2            MOVZBL  #EVL_OBJ,R8                     ; Object number of EVL
          FC48'   30    03B5            BSBW    NET$STARTUP_OBJ                 ; Create EVL process
                        03B8                                                    ; ....ignore any errors
                  05    03B8            RSB
```

```
                          03B9          .SBTTL  Event logging database changes
                          03B9   ;+
                          03B9   ; NET$DBC_ESI - note the receiver database changed
                          03B9   ; NET$DBC_EFI - note the xmitter database changed
                          03B9   ;
                          03B9   ; INPUTS:       NONE
                          03B9   ;
                          03B9   ; OUTPUTS:      R0      Low bit set
                          03B9   ;
                          03B9   ;               All other registers are preserved
                          03B9   ;
                          03B9   ;-
                          03B9   NET$DBC_EFI::
          0FFE 8F    BB   03B9          PUSHR   #^M<R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
            EC       10   03BD          BSBB    STARTUP_EVL              ; Startup EVL if needed
     58   003E'CF    9E   03BF          MOVAB   DBC_EVENT,R8             ; Point to event buffer
     57      68      3C   03C4          MOVZWL  (R8),R7                  ; Get length of item
            FF0C     30   03C7          BSBW    INTERNAL_EVENT           ; Inform EVL of EFI database change
     52   0044 8F    3C   03CA          MOVZWL  #MSG$_EVTXMTCHG,R2       ; This is the mailbox message code
     53      08      D0   03CF          MOVL    #<1@MBX$V_EVTXMTCHG>,R3  ; Set mask
             0C      11   03D2          BRB     DBC_COMMON               ; Finish in common code
                          03D4
                          03D4   NET$DBC_ESI::
          0FFE 8F    BB   03D4          PUSHR   #^M<R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
             D1      10   03D8          BSBB    STARTUP_EVL              ; Startup EVL if needed
     52      3F      3C   03DA          MOVZWL  #MSG$_EVTRCVCHG,R2       ; This is the mailbox message code
     53      04      D0   03DD          MOVL    #<1@MBX$V_EVTRCVCHG>,R3  ; Set mask
                          03E0
                          03E0   DBC_COMMON:
             08      10   03E0          BSBB    BROADCAST                ; Broadcast the message
          0FFE 8F    BA   03E2          POPR    #^M<R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
     50      01      D0   03E6          MOVL    #1,R0                    ; Always successful
                     05   03E9          RSB                              ; Done
                          03EA   ;+
                          03EA   ;
                          03EA   ; BROADCAST - broadcast event related message
                          03EA   ;
                          03EA   ; INPUTS:
                          03EA   ;               R2 - MSG$ code
                          03EA   ;               R3 - mask bit for mailbox filtering   !*** not yet used
                          03EA   ;
                          03EA   ;-
                          03EA   BROADCAST:
     55   0000'CF    D0   03EA          MOVL    NET$GL_PTR_UCB0,R5       ; Point to a NET UCB address
             54      D4   03EF          CLRL    R4                       ; No message text
     50      0A      D0   03F1          MOVL    #NETUPD$_BRDCST,R0       ; Function is "broadcast"
           FC09'     30   03F4          BSBW    CALL_NETDRIVER           ; Call driver comm routine
                     05   03F7          RSB                              ; Done
```

```
                                  03F8        .SBTTL  Outbound raw event processing
                                  03F8    ;+
                                  03F8    ; NET$READ_EVENT - Read out event buffer
                                  03F8    ;
                                  03F8    ; FUNCTIONAL DESCRIPTION:
                                  03F8    ;
                                  03F8    ;       This routine places the event buffer into the specified result (P4)
                                  03F8    ;       buffer. Flags, pointers, and thresholds are reset for more logging.
                                  03F8    ;
                                  03F8    ; INPUTS:       NET$GL_PTR_P3 - Address of result length word
                                  03F8    ;               NET$GL_PTR_P4 - Address of result buffer
                                  03F8    ;               NET$GL_SIZ_P4 - Size of result buffer
                                  03F8    ;
                                  03F8    ; OUTPUTS:      P3, P4 have length of result buffer and result buffer
                                  03F8    ;               R0 - low word has status (NORMAL); high word has byte count
                                  03F8    ;
                                  03F8    ;-
                                  03F8    NET$READ_EVENT::                      ; Entry
            0018'CF      C3       03F8            SUBL3   EVT_L_BUFFER,-        ; Compute no. of bytes in buffer
      50    001C'CF               03FC                    EVT_L_BUFPTR,R0
            56    50     D0       0400            MOVL    R0,R6                 ; Save bfr lth
      57    0000'CF      D0       0403            MOVL    NET$GL_SIZ_P4,R7      ; Get size of result buffer
            56    57     D1       0408            CMPL    R7,R6                 ; Is there room in the result bfr?
                  1C     1E       040B            BGEQU   10$                   ; If GEQU yes
                                  040D            ;
                                  040D            ;   There's not enough room in the caller's buffer to hold
                                  040D            ;   all the events we have buffered.  Scan our event buffer
                                  040D            ;   and find the last event that will fit, so that we always
                                  040D            ;   copy "whole" event records.
                                  040D            ;
      51    0018'CF      D0       040D            MOVL    EVT_L_BUFFER,R1       ; Get buffer pointer
            52    57     D0       0412            MOVL    R7,R2                 ; Get length of user's buffer
            53    61     3C       0415    5$:     MOVZWL  (R1),R3               ; Get length of next event
            52    53     D1       0418            CMPL    R3,R2                 ; Will next event fit in buffer?
                  08     1A       041B            BGTRU   8$                    ; If not, then stop here
            52    53     C2       041D            SUBL    R3,R2                 ; If it fits, then include it
            51    53     C0       0420            ADDL    R3,R1                 ; Skip to next event
                  F0     11       0423            BRB     5$                    ; Keep scanning
      50    57    52     C3       0425    8$:     SUBL3   R2,R7,R0              ; Compute size of bytes to move
                                  0429            ;
                                  0429            ;   The number of bytes to be moved has been determined.
                                  0429            ;   Store the byte count in the P3 result length word.
                                  0429            ;
      51    0000'CF      D0       0429    10$:    MOVL    NET$GL_PTR_P3,R1      ; Get address of result length
                  03     13       042E            BEQL    20$                   ; If EQL there is none
            61    50     B0       0430            MOVW    R0,(R1)               ; Store the result length
                                  0433            ;
                                  0433            ;   Construct the final IOSB with the byte count
                                  0433            ;
            7E    50     B0       0433    20$:    MOVW    R0,-(SP)              ; Byte count to high word
            7E    00'    B0       0436            MOVW    S^#SS$_NORMAL,-(SP)   ; Store I/O status in low word
                                  0439            ;
                                  0439            ;   Move the events into the caller's buffer
                                  0439            ;
0018'DF    02 AE         28       0439            MOVC3   2(SP),@EVT_L_BUFFER,- ; Move event buffer to result bfr
           0000'DF                043F                    @NET$GL_PTR_P4
                                  0442            ;
```

```
                           0442            ;   Shift down any remaining events that couldn't be copied
                           0442            ;   to the front of the buffer.
                           0442            ;
         52   02 AE    3C  0442            MOVZWL  2(SP),R2                  ; Get the number of bytes we moved
      50   56     52   C3  0446            SUBL3   R2,R6,R0                  ; Compute # bytes of remaining events
         001C'CF  52   C2  044A            SUBL    R2,EVT_L_BUFPTR           ; Adjust buffer pointer
 0018'DF   61     50   28  044F            MOVC3   R0,(R1),@EVT_L_BUFFER     ; Move remaining evts to bfr top
 0011'CF   000F'CF     B1  0455            CMPW    EVT_W_LOST,EVT_W_PEAK     ; Did we hit peak "lost bytes"?
                  07   1B  045C            BLEQU   30$                       ; Branch if not
 0011'CF    000F'CF    B0  045E            MOVW    EVT_W_LOST,EVT_W_PEAK     ; Store new peak "lost bytes"
            000F'CF    B4  0465    30$:    CLRW    EVT_W_LOST                ; Clear lost count statistic
                           0469            CLRBIT  #EVT$V_LOSTEVENT,-        ; There's now room in the buffer
                           0469                    EVT_B_FLAGS               ;
         56   02 AE    B1  046F            CMPW    2(SP),R6                  ; Did we empty the buffer?
                  08   13  0473            BEQL    50$                       ; If so, then indicate buffer empty
                           0475            SETBIT  #EVT$V_EVTAVL,EVT_B_FLAGS ; Tell EVL to read more events
                  08   11  047B            BRB     100$                      ; Proceed
            000C'CF    94  047D    50$:    CLRB    EVT_B_FLAGS               ; Reset the flags
            000D'CF    B4  0481            CLRW    EVT_W_THRESH              ; Reset the event threshold
               50   8ED0  0485    100$:   POPL    R0                        ; Get 1st IOSB longword
                       05  0488            RSB                               ; Done
```

```
                              0489          .SBTTL  NET$SET_CTR_TIMER - Reset automatic counter timer
                              0489  ;+
                              0489  ; NET$SET_CTR_TIMER -  Reset automatic counter timer
                              0489  ;
                              0489  ; FUNCTIONAL DESCRIPTION
                              0489  ;
                              0489  ; This routine is called whenever the a data base is updated to start or
                              0489  ; reset the automatic counter timer.  When the counter timer fires, the
                              0489  ; counters will be logged on whatever CNFs are due.  The timer is then
                              0489  ; reset to the next earliest due time.
                              0489  ;
                              0489  ; Inputs:
                              0489  ;
                              0489  ;       R11 = CNR address
                              0489  ;       R10 = CNF address
                              0489  ;
                              0489  ; Outputs:
                              0489  ;
                              0489  ;       None
                              0489  ;
                              0489  ;       R0-R9 are destroyed.
                              0489  ;-
                              0489
                              0489  NET$SET_CTR_TIMER::                         ; Reset logging counter timer
         56   0018'CF    9E   0489          MOVAB   CNX_CRI,R6                  ; Assume CRI data base
       0000'CF     5B    D1   048E          CMPL    R11,NET$GL_CNR_CRI          ; Is it ?
                   19    13   0493          BEQL    10$                         ; If EQL then yes
         56   0030'CF    9E   0495          MOVAB   CNX_NDI,R6                  ; Assume NDI data base
       0000'CF     5B    D1   049A          CMPL    R11,NET$GL_CNR_NDI          ; Is it the NDI data base
                   0D    13   049F          BEQL    10$                         ; If EQL then yes
         56   0000'CF    9E   04A1          MOVAB   CNX_PLI,R6                  ; Assume PLI data base
       0000'CF     5B    D1   04A6          CMPL    R11,NET$GL_CNR_PLI          ; Is it?
                   01    13   04AB          BEQL    10$                         ; Branch if so
                         05   04AD          RSB                                 ; Else, unsupported database
                              04AE
                              04AE          ;
                              04AE          ;   Since it is common for many CNF blocks to be updated by the
                              04AE          ;   network manager at the same time, it is possible to reduce the
                              04AE          ;   total amount of work to be done somewhat by waiting a short time,
                              04AE          ;   the so called "suppression interval", before running the timer
                              04AE          ;   update algorithm after any given CNF block is updated.  This has
                              04AE          ;   the effect of batching the requests and reduces the work by making
                              04AE          ;   better use of each scan of the data base.
                              04AE          ;
                              04AE          ;   The suppression timer interval is 2 seconds.  This is long enough
                              04AE          ;   for a typical  NCP>SET KNOWN NODES ALL  command to complete, and
                              04AE          ;   short enough not to be noticed by the issuer of the command.
                              04AE          ;
         59     08 A6    D0   04AE  10$:    MOVL    CNX$L_DEL_TIME(R6),R9       ; Get the counter timer field i.d.
                FB4B'    30   04B2          BSBW    CNF$GET_FIELD               ; Get its value
                1D 50    E9   04B5          BLBC    R0,15$                      ; If LBC then its not set
  58    00000000'GF      C0   04B8          ADDL    G^EXE$GL_ABSTIM,R8          ; Convert to absolute time
         59     0C A6    D0   04BF          MOVL    CNX$L_ABS_TIME(R6),R9       ; Get field i.d.
                FB3A'    30   04C3          BSBW    CNF$PUT_FIELD               ; Store it
         50     01 A6    9A   04C6          MOVZBL  CNX$B_TIM_SUP(R6),R0        ; Get the suppression timer bit no.
   05 000C'CF      50    E2   04CA          BBSS    R0,EVT_B_FLAGS,15$          ; If BS then update suppression timer
                              04D0                                              ; is ticking
```

```
           58    02   D0 04D0       MOVL    #2,R8                   ; Suppress processing request for 2 sec
                 71   11 04D3       BRB     40$                     ; Set the timer
               0086   31 04D5  15$: BRW     50$                     ; Continue
                         04D8
                         04D8  20$:
                         04D8           ;
                         04D8           ;   Entry point called when timer fires.
                         04D8           ;
                         04D8           ;   Determine database
                         04D8           ;
           50    55   D0 04D8       MOVL    R5,R0                   ; Get the timer WQE for deallocation
     55    51    10   EF 04DB       EXTZV   #16,#16,R1,R5           ; Get timer database i.d.
              FB1D'   30 04E0       BSBW    NET$DEALLOCATE          ; Deallocate WQE
           56 0018'CF 9E 04E3       MOVAB   CNX_CRI,R6              ; Assume CRI timer
           03    55   B1 04E8       CMPW    R5,#EVC$C_SRC_CIR       ; Is it?
                 18   13 04EB       BEQL    25$                     ; If EQL yes
           56 0030'CF 9E 04ED       MOVAB   CNX_NDI,R6              ; Assume NDI timer
           00    55   B1 04F2       CMPW    R5,#EVC$C_SRC_NOD       ; Is it?
                 0E   13 04F5       BEQL    25$                     ; If EQL yes
           56 0000'CF 9E 04F7       MOVAB   CNX_PLI,R6              ; Assume PLI timer
           01    55   B1 04FC       CMPW    R5,#EVC$C_SRC_LIN       ; Is it?
                 04   13 04FF       BEQL    25$                     ; Branch if so
                         0501
                         0501       BUG_CHECK  NETNOSTATE,FATAL     ; Timer i.d. unknown
                         0505
           5B    14 B6 D0 0505 25$: MOVL    @CNX$L_CNR_PTR(R6),R11  ; Get the CNR pointer
           50    01 A6 9A 0509      MOVZBL  CNX$B_TIM_SUP(R6),R0    ; Get the suppression timer bit no.
                         050D       CLRBIT  R0,EVT_B_FLAGS          ; Suppression timer no longer ticking
                 4A   10 0513       BSBB    TICK                    ; Process CNF timers
                         0515           ;
                         0515           ;   Determine the next earliest CNF due time
                         0515           ;
                 5A   D4 0515       CLRL    R10                     ; Start from the head of the CNF list
           59    0C A6 D0 0517      MOVL    CNX$L_ABS_TIME(R6),R9   ; Get absolute time field i.d.
           51    04   D0 051B       MOVL    #NFB$C_OP_FNDMIN,R1     ; Fct is "find minimum value"
              FADF'   30 051E       BSBW    CNF$KEY_SEARCH          ; Find minimum value
              3A 50   E9 0521       BLBC    R0,50$                  ; If no CNF found, no timers are set
              FAD9'   30 0524       BSBW    CNF$GET_FIELD           ; Get due time of minimum CNF
              34 50   E9 0527       BLBC    R0,50$                  ; Branch if cannot get it
        10 B6    58   D0 052A       MOVL    R8,@CNX$L_OLD_TIME(R6)  ; Store the absolute due time
     58   00000000'GF D1 052E       CMPL    G^EXE$GL_ABSTIM,R8      ; Have we passed that time yet?
                         0535                                       ; (this could happen if the event
                         0535                                       ;  buffer is full)
                 05   1F 0535       BLSSU   35$                     ; If LSSU then no
           58    02   D0 0537       MOVL    #2,R8                   ; Try again in 2 seconds
                 0A   11 053A       BRB     40$                     ; Continue
     58   00000000'GF C2 053C  35$: SUBL    G^EXE$GL_ABSTIM,R8      ; Convert to delta time
           58    02   C0 0543       ADDL    #2,R8                   ; CNF timers are grouped into 2 second
                         0546                                       ; buckets to batch the work
                         0546  40$:
                         0546           ;
                         0546           ;   Reset the timer
                         0546           ;
 53   00   00989680 8F  58 7A 0546  EMUL    R8,#10*1000*1000,#0,R3  ; Get quadword timer interval
           52    86 AF 9E 054F      MOVAB   20$,R2                  ; Setup timer routine address
           51    66   D0 0553       MOVL    CNX$W_ID_CTM-2(R6),R1   ; Setup timer i.d. in high order word
        51  0300 8F   B0 0556       MOVW    #WQE$C_QUAL_CTM@8,R1    ; Setup timer qualifier
              FAA2'   30 055B       BSBW    WQE$RESET_TIM           ; Reset the counter timer
```

```
                              05  055E 50$:    RSB
                                  055F
                                  055F
                                  055F
                                  055F
           00000000'GF        D0  055F TICK:   MOVL    G^EXE$GL_ABSTIM,-         ; Get seconds since boot to be used
                0013'CF           0565                 BASE_TIME                 ; as the common base for updating timers
                     5A       D4  0568         CLRL    R10                       ; Start from the head of the CNF list
                                  056A 10$:    ;
                                  056A         ;    Find the next CNF whose timer is due.   Must first check for
                                  056A         ;    CNF entries whose time is past due to prevent finding the same
                                  056A         ;    CNFs over and over again when there are more entries than can fit
                                  056A         ;    in the event buffer.
                                  056A         ;
  50   001C'CF    0018'CF     C3  056A         SUBL3   EVT_L_BUFFER,EVT_L_BUFPTR,R0    ; Compute # bytes in use
  50    00001F00 8F    50     C3  0572         SUBL3   R0,#NET$C_AVLBUF[TR,R0    ; Compute # bytes left
             50    0064 8F    B1  057A         CMPW    #100,R0                   ;&Enough room in buffer?
                     6E       1A  057F         BGTRU   40$                       ; If GTRU then no
              59    0C A6     D0  0581         MOVL    CNX$L_ABS_TIME(R6),R9     ; Get field i.d
        58    10 B6    04     C1  0585         ADDL3   #4,@CNX$L_OLD_TIME(R6),R8 ; Get due time of oldest CNFs
        00000000'GF    58     D1  058A         CMPL    R8,G^EXE$GL_ABSTIM        ; Use 4 second interval but don't
                     0B       1A  0591         BGTRU   13$                       ; exceed the current time
              51     01       D0  0593         MOVL    S^#NFB$C_OP_GTRU,R1       ; Match on key value GTRU CNF field
                  FA67'       30  0596         BSBW    CNF$KEY_SEARCH            ; Find Appropriate CNF
                  13 50       E8  0599         BLBS    R0,15$                    ; If LBS then found one
                     5A       D4  059C         CLRL    R10                       ; Start next scan from head of CNF list
  58    00000000'GF    01     C1  059E 13$:    ADDL3   #1,G^EXE$GL_ABSTIM,R8     ; Bias current time.  The "+1" is used
                                  05A6                                           ; to help smooth the coarseness of the
                                  05A6                                           ; timer and to amortize the timer over-
                                  05A6                                           ; head across a number of CNFs.
              51     01       D0  05A6         MOVL    S^#NFB$C_OP_GTRU,R1       ; Match on key value GTRU CNF field
                  FA54'       30  05A9         BSBW    CNF$KEY_SEARCH            ; Find Appropriate CNF
                  40 50       E9  05AC         BLBC    R0,40$                    ; If LBC then no such CNF
          50    0000'DF       0F  05AF 15$:    REMQUE  @NET$GQ_TMP_BUF,R0        ; Drain temp buffer queue since
                                  05B4                                           ; the search below could fill it
                     05       1D  05B4         BVS     17$                       ; If VS then none
                  FA47'       30  05B6         BSBW    NET$DEALLOCATE            ; Deallocate the block
                     F4       11  05B9         BRB     15$                       ; Loop
                                  05BB 17$:    ;
                                  05BB         ;    Snapshot the counters and log the event records.  The CNT
                                  05BB         ;    action routine will log the event record because the CLRCNT
                                  05BB         ;    flag is set.
                                  05BB         ;
                0000'CF       DD  05BB         PUSHL   NET$GL_FLAGS              ; Save current flags
                                  05BF         SETBIT  #NET$V_CLRCNT,NET$GL_FLAGS ; Counters are to be cleared
          59    04 A6         D0  05C5         MOVL    CNX$L_COUNTER(R6),R9      ; Get counter field i.d.
                  FA34'       30  05C9         BSBW    CNF$GET_FIELD             ; Read/clear the counters
                0000'CF     8ED0  05CC         POPL    NET$GL_FLAGS              ; Restore flags
                                  05D1         ;
                                  05D1         ;    Calculate its next due time and store it in the CNF
                                  05D1         ;
          59    08 A6         D0  05D1         MOVL    CNX$L_DEL_TIME(R6),R9     ; Get delta time field i.d.
                  FA28'       30  05D5         BSBW    CNF$GET_FIELD             ; Fetch it
          59    0C A6         D0  05D8         MOVL    CNX$L_ABS_TIME(R6),R9     ; Get absolute time field i.d.
                  05 50       E8  05DC         BLBS    R0,20$                    ; If LBS then delta time was valid
                  FA1E'       30  05DF         BSBW    CNF$CLR_FIELD             ; Else the timer has been cancelled
                     08       11  05E2         BRB     30$                       ; Continue
```

```
                 58   0013'CF   C0  05E4  1100 20$:     ADDL    BASE_TIME,R8            ; Determine when timer should fire next
                         FA14'  30  05E9  1101          BSBW    CNF$PUT_FIELD           ; Store it
                         FF7B   31  05EC  1102 30$:     BRW     10$                     ; Loop
                                05  05EF  1103 40$:     RSB
                                    05F0  1104
                                    05F0  1105
                                    05F0  1106 .END
```

```
ABS                              000001AB R      04        EVC$C_DLL_LSC                  = 00000140
ACH                              0000016A R      04        EVC$C_DLL_PNEW                 = 00000001
ACP$C_STA_F                    = 00000004                  EVC$C_DLL_POLD                 = 00000000
ACP$C_STA_H                    = 00000005                  EVC$C_DLL_RSC                  = 00000141
ACP$C_STA_I                    = 00000000                  EVC$C_NMA_ABS                  = 00000007
ACP$C_STA_N                    = 00000001                  EVC$C_NMA_CTR                  = 00000008
ACP$C_STA_R                    = 00000002                  EVC$C_NMA_LOS                  = 00000000
ACP$C_STA_S                    = 00000003                  EVC$C_NMA_PRSN                 = 00000003
ADJ$W_PNA                      = 00000004                  EVC$C_NMA_ZER                  = 00000009
BASE_TIME                        00000013 R      02        EVC$C_NSL_DBR                  = 000000C2
BIT...                         = 00000006                  EVC$C_SCL_LNS                  = 00000080
BROADCAST                        000003EA R      04        EVC$C_SCL_PNEW                 = 00000002
BUG$_NETNOSTATE                  ********   X    04        EVC$C_SCL_POLD                 = 00000001
CALL_NETDRIVER                   ********   X    04        EVC$C_SCL_PRSN                 = 00000000
CD1                              00000170 R      04        EVC$C_SRC_ARE                  = 00000005
CD1_2                            00000192 R      04        EVC$C_SRC_CIR                  = 00000003
CIR_ADJ                          00000148 R      04        EVC$C_SRC_LIN                  = 00000001
CIR_BEG                          000000E4 R      04        EVC$C_SRC_NOD                  = 00000000
CIR_COU                          000000B8 R      04        EVC$C_SRC_NON                  = 000000FF
CIR_PKT                          000000D8 R      04        EVC$C_TPL_ACH                  = 00000111
CIR_REASON                       00000151 R      04        EVC$C_TPL_APL                  = 00000100
CNF$CLR_FIELD                    ********   X    04        EVC$C_TPL_ARJ                  = 00000110
CNF$GET_FIELD                    ********   X    04        EVC$C_TPL_AUP                  = 0000010F
CNF$KEY_SEARCH                   ********   X    04        EVC$C_TPL_ILF                  = 0000010B
CNF$PUT_FIELD                    ********   X    04        EVC$C_TPL_IOF                  = 0000010D
CNF$_ADVANCE                   = 00000000                  EVC$C_TPL_ISF                  = 0000010C
CNF$_QUIT                      = 00000002                  EVC$C_TPL_LDF                  = 00000107
CNF$_TAKE_CURR                 = 00000003                  EVC$C_TPL_LDO                  = 00000113
CNF$_TAKE_PREV                 = 00000001                  EVC$C_TPL_LDS                  = 00000112
CNX$B_SPARE                    = 00000000                  EVC$C_TPL_LUP                  = 0000010A
CNX$B_TIM_SUP                  = 00000001                  EVC$C_TPL_OPL                  = 00000103
CNX$C_LENGTH                   = 00000018                  EVC$C_TPL_PADJ                 = 00000008
CNX$L_ABS_TIME                 = 0000000C                  EVC$C_TPL_PFM                  = 00000104
CNX$L_CNR_PTR                  = 00000014                  EVC$C_TPL_PHIA                 = 00000002
CNX$L_COUNTER                  = 00000004                  EVC$C_TPL_PNOD                 = 00000003
CNX$L_DEL_TIME                 = 00000008                  EVC$C_TPL_PPKB                 = 00000001
CNX$L_OLD_TIME                 = 00000010                  EVC$C_TPL_PPKH                 = 00000000
CNX$W_ID_CTM                   = 00000002                  EVC$C_TPL_PRSN                 = 00000005
CNX_CRI                          00000018 R      03        EVC$C_TPL_PRU                  = 00000105
CNX_CRI_OLDTIM                   00000004 R      02        EVC$C_TPL_PSTS                 = 00000007
CNX_NDI                          00000030 R      03        EVC$C_TPL_PVRS                 = 00000006
CNX_NDI_OLDTIM                   00000008 R      02        EVC$C_TPL_RCH                  = 0000010E
CNX_PLI                          00000000 R      03        EVC$C_TPL_RPL                  = 00000102
CNX_PLI_OLDTIM                   00000000 R      02        EVC$C_TPL_UPL                  = 00000101
COU                              000000C0 R      04        EVC$C_TPL_VFR                  = 00000106
COUNTER                          000000A0 R      04        EVC$C_VMS_DBC                  = 00002000
DBC_COMMON                       000003E0 R      04        EVL_OBJ                        = 0000001A
DBC_EVENT                        0000003E R      02        EVT$M_CST_CRI                  = 00000010
ENTER_NO_SRC                     000001B3 R      04        EVT$M_CST_NDI                  = 00000020
ENTER_PKTHDR                     00000220 R      04        EVT$M_CST_PLI                  = 00000008
ENTER_PPKB                       00000249 R      04        EVT$M_DBCEVENT                 = 00000004
ENTER_SRCAREA                    000001BB R      04        EVT$M_EVTAVL                   = 00000001
ENTER_SRCCIR                     000001F5 R      04        EVT$M_LOSTEVENT                = 00000002
ENTER_SRCLIN                     00000214 R      04        EVT$S_CST_CRI                  = 00000001
ENTER_SRCNOD                     000001D1 R      04        EVT$S_CST_NDI                  = 00000001
ENT_17                           00000202 R      04        EVT$S_CST_PLI                  = 00000001
ENT_SRC                          000001FF R      04        EVT$S_DBCEVENT                 = 00000001
```

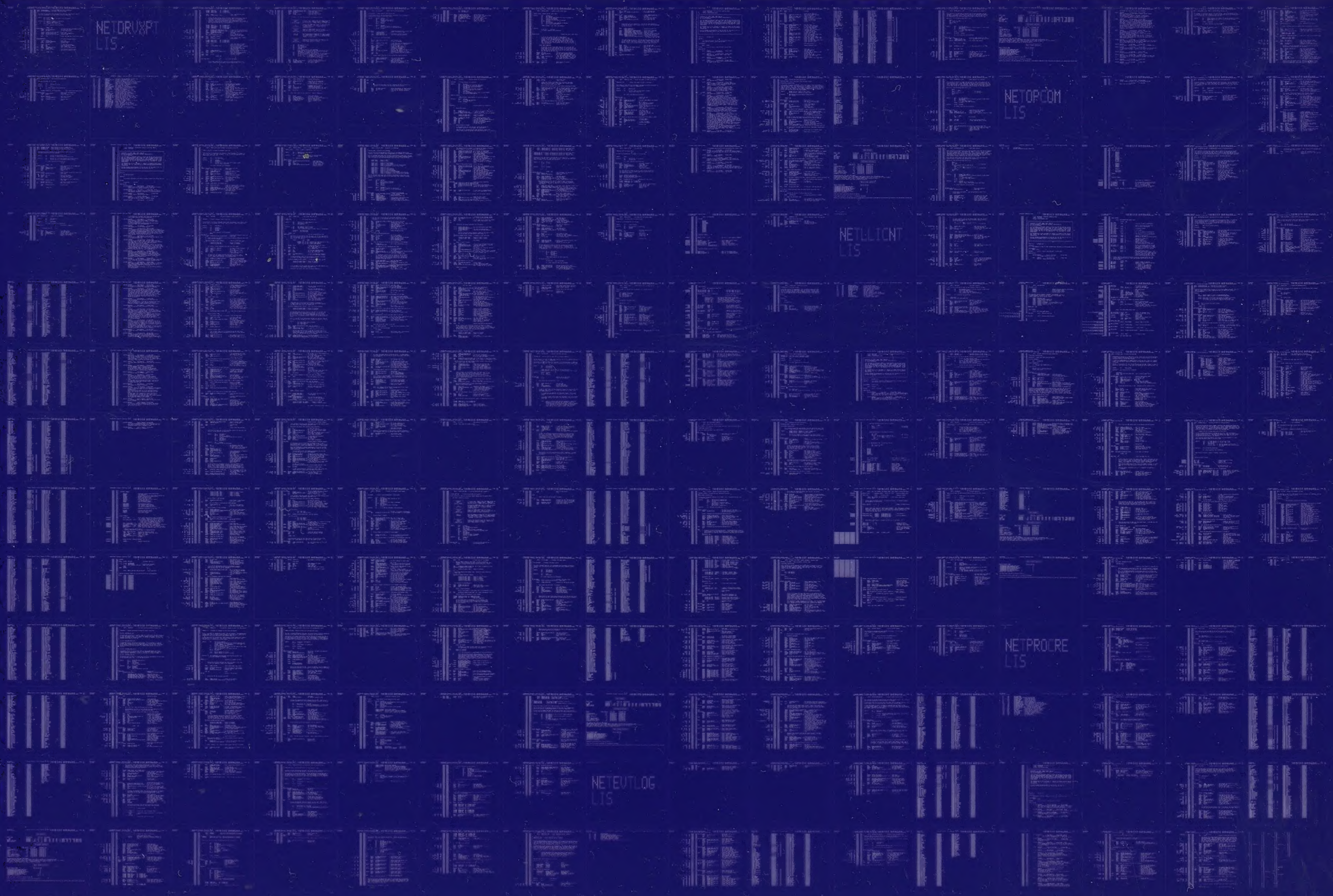
```
EVT$S_EVTAVL                   = 00000001               NET$C_MINBUFSIZ                = 000000C0
EVT$S_LOSTEVENT                = 00000001               NET$C_TID_ACT                  = 00000003
EVT$V_CST_CRI                  = 00000004               NET$C_TID_RUS                  = 00000001
EVT$V_CST_NDI                  = 00000005               NET$C_TID_XRT                  = 00000002
EVT$V_CST_PLI                  = 00000003               NET$C_TRCTL_CEL                = 00000002
EVT$V_DBCEVENT                 = 00000002               NET$C_TRCTL_OVR                = 00000005
EVT$V_EVTAVL                   = 00000000               NET$C_UTLBUFSIZ                = 00001000
EVT$V_LOSTEVENT                = 00000001               NET$DBC_EFI                      000003B9 RG     04
EVT_B_FLAGS                      0000000C R      02     NET$DBC_ESI                      000003D4 RG     04
EVT_L_BUFFER                     00000018 R      02     NET$DEALLOCATE                   ********  X     04
EVT_L_BUFPTR                     0000001C R      02     NET$EVT_INTRAW                   00000017 RG     04
EVT_TIMER                        00000000 R      04     NET$FIND_ADJ                     ********  X     04
EVT_W_LOST                       0000000F R      02     NET$GETUTLBUF                    ********  X     04
EVT_W_PEAK                       00000011 R      02     NET$GL_CNR_CRI                   ********  X     03
EVT_W_THRESH                     0000000D R      02     NET$GL_CNR_EFI                   ********  X     04
EXE$GL_ABSTIM                    ********  X     04     NET$GL_CNR_NDI                   ********  X     03
EXE$GQ_SYSTIME                   ********  X     04     NET$GL_CNR_PLI                   ********  X     03
GET_NDI                          000002A5 R      04     NET$GL_FLAGS                     ********  X     04
ILF                              00000151 R      04     NET$GL_INITVER                   ********  X     04
INTERNAL_EVENT                   000002D6 R      04     NET$GL_PTR_P2                    ********  X     04
IOF                              00000113 R      04     NET$GL_PTR_P3                    ********  X     04
ISF                              00000138 R      04     NET$GL_PTR_P4                    ********  X     04
LDF                              00000151 R      04     NET$GL_PTR_UCB0                  ********  X     04
LDO                              0000013D R      04     NET$GL_PTR_VCB                   ********  X     04
LDS                              0000013D R      04     NET$GL_SIZ_P2                    ********  X     04
LIN_COU                          000000BD R      04     NET$GL_SIZ_P4                    ********  X     04
LNS                              0000017C R      04     NET$GL_UTLBUF                    ********  X     04
LOST_EVENT                       00000020 R      02     NET$GQ_TMP_BUF                   ********  X     04
LSC                              0000019E R      04     NET$LOG_EVENT                    000002CC RG     04
MBX$V_EVTAVL                   = 00000001               NET$M_MAXLNKMSK                = 000003FF
MBX$V_EVTRCVCHG                = 00000002               NET$NDI_BY_ADD                   ********  X     04
MBX$V_EVTXMTCHG                = 00000003               NET$READ_EVENT                   000003F8 RG     04
MSG$_EVTAVL                    = 0000003E               NET$SET_CTR_TIMER                00000489 RG     04
MSG$_EVTRCVCHG                 = 0000003F               NET$STARTUP_OBJ                  ********  X     04
MSG$_EVTXMTCHG                 = 00000044               NET$V_CLRCNT                   = 00000002
NET$AB_EVT_WQE                   0000005C RG     02     NETUPD$_BRDCST                 = 0000000A
NET$ALLOCATE                     ********  X     04     NFB$C_CRI_CNT                  = 04020044
NET$C_ACT_TIMER                = 0000001E               NFB$C_CRI_CTA                  = 04010011
NET$C_AVLBUFLTH                = 00001F00               NFB$C_CRI_LCT                  = 04010015
NET$C_EFN_ASYN                 = 00000002               NFB$C_CRI_NAM                  = 04020041
NET$C_EFN_WAIT                 = 00000001               NFB$C_NDI_CNT                  = 02020042
NET$C_EVTBUFLTH                = 00001F40               NFB$C_NDI_CTA                  = 02010011
NET$C_EVTTHRESH                = 00000005               NFB$C_NDI_CTI                  = 02010013
NET$C_EVTTIMER                 = 02FAF080               NFB$C_NDI_NNA                  = 02020043
NET$C_IPL                      = 00000008               NFB$C_OP_FNDMIN                = 00000004
NET$C_LSTEVTLTH                = 00000020               NFB$C_OP_GTRU                  = 00000001
NET$C_MAXACCFLD                = 00000027               NFB$C_PLI_CNT                  = 05020044
NET$C_MAXLINNAM                = 0000000F               NFB$C_PLI_CTA                  = 05010010
NET$C_MAXLNK                   = 000003FF               NFB$C_PLI_LCT                  = 05010013
NET$C_MAXNODNAM                = 00000006               NFB$C_PLI_NAM                  = 05020041
NET$C_MAXOBJNAM                = 0000000C               NMA$C_PTY_AI                   = 00000040
NET$C_MAX_AREAS                = 0000003F               NMA$C_PTY_CD1                  = 00000081
NET$C_MAX_LINES                = 00000040               NMA$C_PTY_CM1                  = 000000C1
NET$C_MAX_NCB                  = 0000006E               NMA$C_PTY_CM2                  = 000000C2
NET$C_MAX_NODES                = 000003FF               NMA$C_PTY_CM3                  = 000000C3
NET$C_MAX_OBJ                  = 000000FF               NMA$C_PTY_CM4                  = 000000C4
NET$C_MAX_WQE                  = 00000014               NMA$C_PTY_DU1                  = 00000001
```

```
NMA$C_PTY_DU2                    = 00000002
NMA$C_PTY_H1                     = 00000021
NMA$C_PTY_HI                     = 00000020
NOD_COU                            000000B3 R       04
NON_PKT                            000000D2 R       04
NSP$C_EXT_LNK                    = 0000001E
NSP$C_MAXHDR                     = 00000009
PNA_NODE                           00000262 R       04
PRU                                000000F0 R       04
RAW$B_SRCTYP                       0000000C
RAW$C_SIZE                         0000001F
RAW$K_SIZE                         0000001F
RAW$T_DATA                         0000001E
RAW$T_SRCID                        0000000D
RAW$T_SYSTIM                       00000002
RAW$W_BYTES                        00000000
RAW$W_EVTCODE                      0000000A
RCB$W_ADDR                       = 0000000E
RCH                                00000162 R       04
RSC                                0000019E R       04
SEND_EVT_MSG                       00000388 R       04
SIZ...                           = 00000001
SS$_BADPARAM                       ********    X    04
SS$_NORMAL                         ********    X    04
STARTUP_EVL                        000003AB R       04
SUPPRESS_AREA                      ********    X    04
TICK                               0000055F R       04
TR$C_MAXHDR                      = 0000001C
TR$C_NI_ALLEND1                  = 040000AB
TR$C_NI_ALLEND2                  = 00000000
TR$C_NI_ALLROU1                  = 030000AB
TR$C_NI_ALLROU2                  = 00000000
TR$C_NI_PREFIX                   = 000400AA
TR$C_NI_PROT                     = 00000360
TR$C_PRI_ECL                     = 0000001F
TR$C_PRI_RTHRU                   = 0000001F
VFR                                0000010A R       04
WQE$B_EVL_DT1                    = 0000001E
WQE$B_EVL_DT2                    = 0000001F
WQE$C_LENGTH                     = 00000024
WQE$C_QUAL_CTM                   = 00000003
WQE$DEALLOCATE                     ********    X    04
WQE$FORK                           ********    X    04
WQE$L_EVL_PKT                    = 00000018
WQE$RESET_TIM                      ********    X    04
WQE$W_ADJ_INX                    = 00000020
WQE$W_EVL_CODE                   = 0000001C
WQE$W_REQIDT                     = 00000012
_$$_                             = 00000000
```

```
                                              +----------------+
                                              ! Psect synopsis !
                                              +----------------+

PSECT name                       Allocation          PSECT No.  Attributes
----------                       ----------          ---------  ----------
.  ABS  .                        00000000  (     0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
$ABS$                            0000001F  (    31.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
NET_IMPURE                       00000080  (   128.)  02 (  2.)  NOPIC   USR   CON   REL   LCL NOSHR NOEXE   RD    WRT NOVEC LONG
NET_PURE                         00000048  (    72.)  03 (  3.)  NOPIC   USR   CON   REL   LCL NOSHR NOEXE   RD  NOWRT NOVEC LONG
NET_CODE                         000005F0  (  1520.)  04 (  4.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD  NOWRT NOVEC LONG

                                           +------------------------+
                                           ! Performance indicators !
                                           +------------------------+

Phase                        Page faults    CPU Time       Elapsed Time
-----                        -----------    --------       ------------
Initialization                        31    00:00:00.07    00:00:00.36
Command processing                   155    00:00:01.01    00:00:04.50
Pass 1                               601    00:00:23.83    00:00:32.66
Symbol table sort                      0    00:00:03.08    00:00:03.16
Pass 2                               257    00:00:04.82    00:00:06.17
Symbol table output                   36    00:00:00.26    00:00:00.26
Psect synopsis output                  2    00:00:00.03    00:00:00.03
Cross-reference output                 0    00:00:00.00    00:00:00.00
Assembler run totals                1084    00:00:33.10    00:00:47.14

The working set limit was 1950 pages.
126517 bytes (248 pages) of virtual memory were used to buffer the intermediate code.
There were 130 pages of symbol table space allocated to hold 2195 non-local and 76 local symbols.
1106 source lines were read in Pass 1, producing 25 object records in Pass 2.
39 pages of virtual memory were used to define 35 macros.

                                          +--------------------------+
                                          ! Macro library statistics !
                                          +--------------------------+

Macro library name                        Macros defined
------------------                        --------------
_$255$DUA28:[SHRLIB]NMALIBRY.MLB;1                     1
_$255$DUA28:[SHRLIB]EVCDEF.MLB;1                       2
_$255$DUA28:[NETACP.OBJ]NETDRV.MLB;1                   0
_$255$DUA28:[NETACP.OBJ]NET.MLB;1                     14
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                         1
_$255$DUA28:[SYSLIB]STARLET.MLB;2                      8
TOTALS (all libraries)                                26

2325 GETS were required to define 26 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:NETEVTLOG/OBJ=OBJ$:NETEVTLOG MSRC$:NETEVTLOG/UPDATE=(ENH$:NETEVTLOG)+EXECML$/LIB+LIB$:NET/LIB+LIB$:NETDRV/LIB+SHRLIB$
```

0278 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



NETDRV%PT
LIS

NETOPCOM
LIS

NETLLICNT
LIS

NETPROCRE
LIS

NETEVTLOG
LIS